Applications of Algebraic Logic and Universal Algebra
to Computer Science
Grant no. N00014-88-J-1182

Department of the Navy
Office of Naval Research, Code 1513:CMB

Final Project Report

Summary

A four-day conference was held at Iowa State University oriented around the topics stated in the project title. Approximately 80 people attended the conference, with roughly equal representation from Mathematics and Computer Science.

The conference consisted of eight invited lectures (60 minutes each) and 26 contributed talks (20-40 minutes each). There was also a "round-table discussion" on the role of algebra and logic in computer science.

The expenses of the invited speakers was partially supported by this grant. The invited talks, and the round-table were recorded on video tape and are available to the public. The proceedings of the conference will be published by Springer-Verlag in its "Lectures Notes in Computer Science" series.

Invited Speakers

Eight invited, one-hour talks lectures were presented. The speakers were:

Eric Wagner	IBM
Joel Berman	University of Illinois at Chicago
Dana Scott	Carnegie-Mellon University
Istvan Nemeti	Hungarian Academy of Sciences
Dexter Kozen	Cornell University
H.P. Gumm	SUNY New Paltz
Vaughan Pratt	Stanford University
Bjarni Jonsson	Vanderbilt University

The titles of the lectures may be gleaned from the enclosed program.

Contributing Speakers

There were 26 contributed talks, ranging in length from 20 to 40 minutes. Rather than list them all here, a copy of the program of the conference is included with this report. In addition, a "round-table discussion" was held one evening. The topic was the role of algebra and logic in computer science. The panel consisted of the invited speakers.

Dissemination of the Proceedings

The proceedings of the conference will be published by Springer-Verlag in the Fall of 1989. We hope to have contributions from all of the invited speakers, as well as a number of contributed papers. Papers are currently being refereed. In addition, the invited talks, and the evening discussion were videotaped. Copies of the tapes have already been sent to ONR. They are available for purchase by the public from Iowa State University.

Enclosures

A copy of the conference program and a set of abstracts of most of the talks is enclosed with this report.

Algebraic Logic and Universal Algebra in Computer Science

Iowa State University

Wednesday, June 1

9:00– 9:30 Registration, conference center
9:30– 9:40 Opening remarks: Norman L. Jacobson
9:40–10:40 Invited Address: Eric Wagner, IBM, *All recursive types defined using products and sums can be implemented using pointers*
10:40–11:00 break
11:00–11:30 Ivo Rosenberg, Univ. Montreal, *Term equations via Mal'cev preiterative algebras*
11:40–12:10 Ildiko Sain, Hung. Acad. Sci., *Comparative study of distinguished program verification methods*

Lunch

1:30– 2:30 Invited Address: Joel Berman, Univ. Illinois, Chicago, *The value of free algebras*
2:30– 2:50 break
2:50– 3:20 Lawrence Moss, Univ. Michigan, *Final algebra semantics for insufficiently complete specifications*
3:30– 4:00 Ivan Rival, University of Ottawa, *Graphical data structures for ordered sets*
4:00– 4:20 break
4:20– 4:40 Hong X. Dang, SUNY, Geneseo, *On sublattice lattice varieties*
4:50– 5:20 George McNulty, Univ. South Carolina, *Avoidable Words*

Dinner

7:00–10:00 Reception (cash bar), Campanile Room, Memorial Union

Thursday, June 2

9:00–10:00 Invited Address: Dana Scott, Carnegie-Mellon, *Domains and algebras*
10:00–10:20 break
10:20–10:50 Zbigniew Stachniak, York Univ., *The resolution rule: an algebraic perspective*
11:00–11:20 Alan Day, Lakehead Univ., *The interval construction revisited*
11:20–11:40 break
11:40–12:00 Ernie Manes, Univ. Massachusetts

Lunch

1:30– 2:30 Invited Address: Istvan Németi, Hungarian Academy of Sciences, *Epimorphisms in algebraic logic with applications to the Beth definability theorem*
2:30– 2:50 break
2:50– 3:10 Marek Zaionc, Univ. of Alabama, Birmingham, *A characteristic of definable tree operations*
3:20– 3:40 H. Albert Lilly, Univ. of Alabama, Birmingham, *A survey of current research into the use of functional specifications for the compilation of programming languages*
3:40– 4:00 break
4:00– 4:30 Hajnal Andréka, Hungarian Academy of Sciences
4:40– 5:10 Richard Thompson, Univ. of California, *The manipulatory foundations of non-parallel programming*

Dinner

7:30– 9:30 Round-table discussion: The role of algebra and logic in computer science. Moderator: George Strawn, ISU, 101 Carver Hall

Friday, June 3

9:00–10:00 Invited Address: Dexter Kozen, Cornell, *Stone duality in programming language semantics*
10:00–10:20 break
10:20–10:40 H.P. Sankappanavar, SUNY New Paltz, *Linked double weak Stone algebras*
10:50–11:10 Erzsébet Lukács, Vanderbilt, *Representability of finite relation algebras with many identity atoms*
11:10–11:30 break
11:30–12:10 Mitsuhiro Okada, Concordia Univ., *Algebraic proof of normalization theorems for polymorphic lambda calculus and higher order logics*

Lunch

1:40– 2:40 Invited Address: H.P. Gumm, *The role of universal algebra in computer science*
2:40– 3:00 break
3:00– 3:30 Fernando Guzman, SUNY Binghamton, *Conditional logic*
3:40– 4:10 George Nelson, Univ. Iowa, *Other logics for equational theories*
4:10– 4:30 break
4:30– 4:50 Robert W. Quackenbush, University of Manitoba, *The completeness theorem for the universal logic of algebras via congruences*
5:00– 5:30 Irving Anellis, Philosophia Mathematica, *Maslov's Inverse Method and its application to programming logic*
7:30 Banquet–Starlight Village Motel

Saturday, June 4

9:00–10:00 Invited Address: Vaughan Pratt, Stanford Univ., *Universal algebra or category theory?*
10:00–10:20 break
10:20–10:50 Marek Suchenek, Wichita State, *Incremental models of incomplete information data bases*
11:00–11:20 Nistala V. Murthy, Univ. of Toledo, *Essentially algebraic categories*
11:20–11:40 break
11:40–12:10 W.D. Maurer, George Washington Univ., *Three fundamental correctness theorems for the modification index method*

Lunch

1:40– 2:40 Invited Address: Bjarni Jónsson, Vanderbilt, *Relatively free relation algebras*
2:40– 3:00 break
3:00– 3:40 David Benson, Washington State Univ., *Interaction automata*
3:50– 4:20 R. Padmanabhan, Univ. of Manitoba, *Equational logic on algebraic curves*
4:20– 4:40 break
4:40– 5:10 Chihyi Ying, *First-order boolean-valued semantics*

General information: All lectures will be held in room 101 Carver Hall. Registration will be in the lobby of Carver Hall on Wednesday morning. The Memorial Union cafeteria will be open on June 1–5 during the following hours:

Wednesday	7 a.m.–2:30 p.m.,	5 p.m.–6:30 p.m.
Thursday	7 a.m.–2:30 p.m.,	5 p.m.–6:30 p.m.
Friday	7 a.m.–2:30 p.m.	
Saturday	7 a.m.–1:15 p.m.	
Sunday	8 a.m.–1:15 p.m.	

Algebraic Logic and Universal Algebra in Computer Science

Wednesday June 1–Saturday June 4, 1988

Abstracts

Abstract

Iowa State University Conference on Algebraic Logic and Universal Algebra in Computer Science

MASLOV'S INVERSE METHOD AND ITS APPLICATION TO PROGRAMMING LOGIC

Irving H. Anellis

Sergei Iurii Maslov first presented his inverse method for establishing derivability in logical calculi in 1964, applying his method to the decidability of special cases of the decision problem. Soviet researchers such as Davydov soon developed Maslov's method for use in machine computations. The first use of Maslov's method in programming logic occurred in 1969, when Davydov and others adapted Maslov's method as a machine algorithm for establishing derivability on the basis of the inverse method.

Maslov's method provides the basis for a theorem-proving program which presents a fully-determined decision procedure for the Maslov class K for all formulae of classical first-order logic, whether these formulae are in prenex form or not, and also to decision problems in group theory and for quadratic and biquadratic equations. As such, Maslovs method is more powerful than Robinson's resolution method.

Maslov also developed an iterative method as a specialized case of the inverse method for deciding the satisfiability of propositional formula. The iterative method is a rationalization of a brute-force search for a path through an integer matrix, and can be shown to be a trivial reformulation of the decision problem for decidability of propositional formulae, a problem which Maslov showed to be NP-complete. A recent (1987) issue of Voprosy Kibernetika was devoted to exclusive consideration of the problems in reduction of the brute-force search procedure, with special reference to Maslov's iterative method.

AMS (MOS) 1980 subject classifications, 1985 revision:
Primary: 03-04, 03B25, 03B35, 03B70, 03F07; 08-04, 15-04, 68-04, 68Q99.
Secondary: 03D05, 03F05, 03F20, 68Q05, 68Q40, 68Q70; 68N05, 68T15.

Interaction Automata

David B. Benson
Computer Science Department
Washington State University
Pullman, WA 99164–1210

Abstract

Interaction automata proceed by synchronized agreement on communication symbols. The automata have both inputs and outputs, which are identified. Thus the automaton actions $S \otimes A \to S$ and the coactions $S \to A \otimes S$ are isomorphic. The theory then requires a *dimonoid* consisting of a commutative monoid multiplication

$$\nabla\colon\ A \otimes A \to A$$

and a cocommutative comonoid comultiplication

$$\Delta\colon\ A \to A \otimes A$$

such that

$$\nabla\Delta = (\Delta \otimes \mathbf{1})(\mathbf{1} \otimes \nabla).$$

See Carboni & Walters, Cartesian Bicategories, *J. Pure Appl. Alg.*, recent.

The talk will motivate this structure in the theory of communicating processes and develop certain consequences. In particular we shall show that a universal communicator with a given interaction automata follows from considerations similar to the Smyth-Plotkin theory for solutions to the domain equation $D \cong [D \to D]$, but requires some elementary considerations of lattice-ordered commutative groups.

If time allows, relationships to Girard's Linear Logic will be developed.

H.P.Gumm , SUNY, New Paltz

The role of universal algebra in computer science

The language of universal algebra and its most fundamental results and methods have become essential modelling tools in key areas of computer science. We shall explain some of its typical uses in the fields of data structures, programming languages, database theory and data encoding. A number of interesting concepts and challenging questions arise from the computer science applications of universal algebra that should be interesting to study from a purely mathematical standpoint too.

Fernando Guzman, SUNY Binghamton

CONDITIONAL LOGIC

ABSTRACT

In many programming languages like C, Prolog and Lisp among others, short-circuit evaluation is used in processiong logic expressions; i.e. the conjunction "e1 and e2" is evaluated left to right and evaluation stops as soon as possible: if e1 is FALSE the whole expression is FALSE, if e1 is TRUE then e2 is evaluated and this result becomes the value of the whole expression. This conditional evaluation of e2 allows expressions like:

x != 0 and y/x < 10

since the second argument of the 'and' is not evaluated when x = 0.

The study of this logic requires the introduction of a third truth value "U" to stand for any or all of the following: 'undefined', 'error' or 'non-terminating evaluation'. This is how we arrive at a three-valued logic, called Conditional Logic. It should be mentioned that languages like Pascal that use Kleene's weak logic for the binary operators 'and' and 'or', also exhibit some form of coditional evaluation in their conditional statements.

Our study of Conditional Logic includes the following algebraic aspects:

- Axiomatization of the variety generated by this 3-element algebra.
- Detailed description of the finitely generated free algebras including their cardinalities.
- A representation theorem for the algebras in this variety, analogous to that of Boolean algebras.

Besides the full Conditional Logic ('and', 'or' and 'not' operators), we'll also look at the two implication algebras that live in it. The existence of two non-isomorphic implication algebras follows from the fact that the 'and' and 'or' operators are not commutative. Thus we get:

$e1 \Longrightarrow_1 e2 = (\text{not } e1) \text{ or } e2$; $e1 \Longrightarrow_2 e2 = e2 \text{ or } (\text{not } e1)$

For each of these two implication algebras we also get the axiomatization of the corresponding variety, and a description of the finitely generated free algebras including their cardinalities.

On Sublattice Lattice Varieties

Dang X. Hong
SUNY College at Geneseo

In *General Lattice Theory* by G. Grätzer, there are listed two problems:

Problem I.6. Determine all equational classes of lattices $\mathcal{K}$ for which $\mathrm{Sub}(\mathcal{K}) = \mathcal{K}$.

Problem I.7. Determine all equational classes of lattices $\mathcal{K}$ for which $\mathrm{Sub}(\mathcal{K})$ = The equational class of all lattices.

We solved these problems by showing that

Theorem. For any non-trivial equational class of lattices $\mathcal{K}$, $\mathrm{Sub}(\mathcal{K})$ is the equational class of all lattices.

Note: $\mathrm{Sub}(\mathcal{K})$ denotes the equational class generated by all lattices of sublattices of members of $\mathcal{K}$.

Duality of Predicate Transformers and Powerdomains

DEXTER KOZEN

Cornell University

In this talk we outline a general theory of duality between powerdomains and predicate transformers. This duality is most clearly illustrated in the framework of probabilistic programs. We argue that the order topology on powerdomains and the concept of algebraicity become less relevant from this point of view.

A SURVEY OF CURRENT RESEARCH INTO THE USE OF FUNCTIONAL SPECIFICATIONS FOR THE COMPILATION OF PROGRAMMING LANGUAGES

H. Albert Lilly
Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294

The advantages of the functional style of programming have been applied to the specification of programming languages for the purpose of studying the compilation process from a mathematical perspective. The specification process demands a critical evaluation of both the variations of languages that are possible inputs to the compiler and the types of transformations that might be used during the compilation. In simple mathematical terms this involves the defining of domains and of functions which map the domains into the desired result. Ideally, the result corresponds to the denotational interpretation of the original specification and the compilation is a proof of that correspondence, i.e. the correctness of the compiler is verified.

A broad research interest is how language features should be mathematically defined and what the implications are for verification of the transformation. The combination of differences in language features creates an infinite number of languages to consider. Each language taken from the infinite set of languages can be thought of as having certain algebraic properties unique to that language. Of interest is how one language compares to another in mathematical terms and how this affects the compilation. A problem that has not been solved in the general case is to how factor out the differences between two languages without necessarily sacrificing the desirable transformational mathematical properties.

Another broad research interest involves determining what the alternative compiler designs are and which is the best for a given language. This area entails the study of combinators used in reduction rules and the usual space/time tradeoffs. A related area in compiler design is the implication of viewing a compiler as an abstract machine and how the semantics of compiler construction can be applied to architectural issues. For example, how should a functional mapping be specified in a parallel environment?

Three Fundamental Correctness Theorems for the Modification Index Method

W.D. Maurer

The George Washington University

We present here a formal justification of the modification index method of proving programs correct. This method has several advantages over the more commonly used back substitution method. Proofs constructed with the modification index method are easy to debug, since each term in the proof is directly related to some statement, assertion or termination expression of the given program. The lengths of verification conditions increase more slowly, as a function of the length of the path, if this method is used. Also, cleanness and termination may be handled in the same way as partial correctness. The statement of the second theorem is of importance because the order in which the terms must be combined is not at all obvious; the statement of the third theorem is of importance because termination expressions, in practice, do not, in any obvious way, take values in a well-ordered set, as required in Floyd's original formulation of this theorem.

Nistala V. Murthy, University of Toledo

ABSTRACT

Several of the constructs that are usually studied in algebra, let us call them ' Algebraic Constructs ' -such as *Grp*, *Mon*, *R-Mod*, *Rng*, , *R-Alg*, *Lat*, *Boolalg* -arise as full subconstructs of the constructs of partial algebras of type τ for various τ, and they have many well known properties in common like

bijective morphisms being isomorphisms

being closed under cartesian products

being closed under subobjects

admitting free objects

admitting initial objects

admitting terminal objects

having equalizers

---

etc.

In other words for a full subconstruct of $ParAlg(\tau)$ to be an algebraic construct it is essential that it has the above mentioned properties. On the other hand consider the following full subconstruct, C, of $ParAlg(1,1)$, suggested by Horst Herrlich, with objects of the form (X,a,b) where a is a total unary operation and b is a partial operation whose domain is equationally determined by $(a,1)$, i.e. $x \in dom(b)$ iff $a(x) = x$.

One can show that C has all the previously mentioned properties and is likely to be called an algebraic construct. It is interesting to know what makes C reflect this ' algebraic nature ' . However, one should notice here that C consists of partial algebras where the domain of each operation is equationally determined by the previous operations (Page 4, "Aspects of Topoi", Bull. Austral. Math. Soc., Vol 7 (1972), 1-76.)

Now if τ is an arbitrary type, is it true that any full subconstruct C of $ParAlg(\tau)$ with objects partial algebras where the domain of each operation is equationally determined by the previous operations -let us call these constructs equationally determined constructs -will still have those well known properties ? The answer is YES. In the first part these equationally determined constructs are precisely defined via polynomial symbols and their realizations, and their properties are investigated.

Now when we come back to our algebraic constructs, we notice that each object in any of these constructs, in addition to being equationally determined, has a ' variety structure ' on it, i.e. certain identities are satisfied. This variety structure on the full subconstructs of $ParAlg(\tau)$ is separately studied in the second part under Equationally Defined Constructs.

In the final part, Essentially Algebraic Categories in the sense of Peter Freyd are defined precisely; these are categories equivalent to the categories associated to intersections of equationally determined constructs and equationally defined constructs.

On the other hand Horst Herrlich defined essentially algebraic categories in his "Essentially Algebraic Categories", Quaestiones Mathematicae, Vol 9 (1986), 245-262. In one proposition we prove that essentially algebraic categories in the sense of Peter Freyd are essentially algebraic which partly answers the problem of Horst Herrlich, namely, What is the relation between essentially algebraic categories and essentially algebraic categories of Peter Freyd ?

OTHER LOGICS FOR EQUATIONAL THEORIES

G. C. Nelson

Let L be a countable algebraic language, i.e., L has countably many function symbols of finite arity together with equality. Let Σ be a set of equations of L and let us denote by P the set of positive formulas of L which is the smallest set of formulas of L containing the atomic formulas and closed under conjunctions $\wedge$, disjunctions $\vee$, and both kinds of quantification $\forall x$ and $\exists x$. We may assume that the language L has only the above logical symbols so that every formula is positive. The rules of inference that we introduce for L are the usual ones first used for the equational logic by Birkhoff which are used to infer equations from other equations and are called substitution, replacement, and properties of equality together with the following: $\frac{\varphi, \psi}{\varphi \wedge \psi}$, $\frac{\varphi}{\forall x \varphi}$, $\frac{\varphi}{\varphi \vee \theta}$, $\frac{\varphi}{\theta \vee \varphi}$, and $\frac{\varphi'^{y}_{t}}{\exists x \varphi}$, where $\exists y \varphi'$ is an alphabetic variant of $\exists x \varphi$, t is a term of L in at most those variables occurring free in $\exists x \varphi$, t is free for y in φ', and φ'^{y}_{t} denotes the result of replacing all free occurrences of y in φ' by t. As is usual, this notation means that one can infer the formula below the line from the formulas above that line. A *positive proof of* φ *from* Σ is any finite sequence of positive formulas whose last element is φ such that each formula in the sequence is either an element of Σ or follows from previous elements in the sequence by one of the above–mentioned rules of inference.

Theorem: For any positive formula φ, φ is true in the variety determined by Σ iff there is a positive proof of φ from Σ.

Let L be a finite algebraic language and $\mathbf{A}$ be a finite algebra for L with k elements. We let $V(\mathbf{A})$ denote the variety generated by $\mathbf{A}$. For a term $t(x_1,\ldots,x_n)$ we denote by $t^{x_j}_{x_i}$ the result of replacing all occurrences of x_j by x_i. For $n \geq k$ we introduce a combinatorial rule of inference denoted by $\frac{\{t^{x_j}_{x_i} = s^{x_j}_{x_i} : 1 \leq i < j \leq n\}}{t = s}$ where t and s are terms in at most the variables $x_1,\ldots,x_n$. Again we allow also the rules of inference called substitution, replacement, and properties of equality and we say an equation $t = s$ is k–*combinatorially provable from* Σ if it is the last equation in a finite sequence of equations each of which is in Σ or follows from previous equations by the rules of inference mentioned in this paragraph.

Proposition: There is a finite set of equations Σ such that an equation is true in $\mathbf{A}$ iff it is k–combinatorially provable from Σ.

A LOGICAL ANALYSIS OF TERM REWRITE SYSTEMS AND COMPUTATION OF EQUATIONAL THEORIES
(Abstract)

Mitsuhiro Okada

Department of Computer Science
Concordia University
Montreal, Canada

The purpose of this paper is to provide new application of proof theory in logic to term rewriting theory in computer science. It is shown that proof theoretic tools are very useful for analysing the basic properties, the termination property and the Church-Rosser property of term rewrite systems. Term Rewriting Systems have been studied in computer science for algebraic specification of abstract data types, equational programming, automated theorem proving, symbolic computation, etc.

Term rewrite systems are logic-free equational systems without symmetric axiom (the axiom of the form $a=b \Rightarrow b=a$). Then equality $a=b$ has an oriented direction and is interpreted as a reduction or a transformation from one term to another. We use the expression $s \rightarrow t$ for a reduction of a term s to a term t.

Our goal is to obtain criteria of equivalence between a given conditional equational system and the corresponding conditional rewrite system, or to make a conditional rewrite system which is equivalent to a given conditional equational system. Here equivalence between a conditional equational system E and a conditional rewrite system R means "$E \vdash s=t$ if and only if $R \vdash s \downarrow t$". $s \downarrow t$ means "there exists a term u such that $s \rightarrow^* u \ ^*\leftarrow t$", and $\rightarrow^*$ is the reflexive-transitive closure of $\rightarrow$. Equality $s=t$ is interpreted in a rewrite sytem as $s \downarrow r_1 \downarrow r_2 \downarrow \ldots \downarrow r_n \downarrow t$ for some terms $r_1, r_2, \ldots, r_n$, which is denoted by $s=^*t$. The reduction of $s=t$ to $s \downarrow t$ means a direct computation of $s=t$. A rewrite system is said to be terminating if there is no infinite sequence of reductions. A rewrite system is said to be Church-Rosser if $s=^*t$ implies $s \downarrow t$ for any term s,t.

In section 1 we give the relationship between the proof theoretic ordinals in logic and the ordering structures used in termination proof and in the Knuth-Bendix completion procedure of term rewriting theory. In section 2 we utilize the proof-theoretic normalization technique to analyze Church-Rosser property and completion procedure for conditional term rewriting theory. In the course of this study, we show that Knuth's critical pair lemma does not hold for conditional rewrite systems, by presenting a counter-example. Then we present two restrictions on conditional system so that the critical pair lemma holds. One is considered a generalization of Bergstra-Klop's former result, the other is concerned with a generalization of Kaplan's and Jouannaud-Waldmann's systems.

Algebraic Proof of Normalization Theorems for Polymorphic Lambda Calculus and Higher Order Logics (Abstract)

Mitsuhiro Okada

Department of Computer Science
Concordia University
Montreal, Canada

In this paper we present a Heyting algebra which is very useful and powerful for the proof of the normalization theorems for various versions of Polymorphically Typed Lambda Calculus. It is also shown that the same algebra is powerful for the proof of the normalization and cut-elimination theorems for a very wide range of logics, including first-order, second-order and higher-order formulations of both classical and intuitionistic (and minimal) logics and their modifications to modal logics. Also this technique does not depend on the difference of the formulation between the sequent calculus (based on Gentzen's LK or LJ or its modified versions like Beth's tableaux) and the natural deduction (based on Gentzen's NK or NJ).

As an example of the use of our Heyting algebra, we first show the normalization theorem for Polymorphically Typed Lambda Calculus, then we show the cut-elimination theorem (and the normalization theorem at the same time) for higher order intuitionistic modal logics based on S4 and T.

We discuss the differences and the relationships between our approach and Girard-Martin-Löf-Prawitz's approach, and between our approach and Tait-Mitchell's approach.

EQUATIONAL LOGIC ON ALGEBRAIC CURVES

If F and G are two algebraic curves of degrees m and n respectively in the complex projective plane with *no common component* then the classical Bezout Theorem says that the number of points common to F and G is mn, counting multiplicities. This situation gives rise to natural universal algebraic structures on cubic curves. For example, let x, y, z, t and u be five distinct points on an irreducible non-singular cubic curve F. Then there exist a unique conic Q passing through the five given points. By Bezout theorem, $|F \cap Q| = 6$ and so we have a partial 5-ary law of composition q(x, y, z, t, u) defined on the cubic by this geometrically motivated 'conic process'. It is known that the operation 'q' is definable on the entire curve F , i.e. $\langle F; q \rangle$ is a universal algebra of type $\langle 5 \rangle$ and, moreover, 'q' is a rational function of the coordinates of the five points. We call these structures Bezout algebras of type $\langle 5 \rangle$ and the operation 'q' as a Bezout law. An algebra thus constructed has purely algebraic, topological and, of course, geometric properties and these various properties blend so well that each has a profound influence on the other two. Consequently, the first order theory of such algebraic systems admit a richer set of rules of derivation. For example, every group operation which is a derived polynomial of a Bezout law must be commutative! The following is a list of formal algebraic principles valid for the first order theory of clones of such laws in addition to the usual five rules of equational logic:

1. distinct validity → global validity
2. local cancellation → global solvability
3. local independence → global independence
4. the overlay technique.

Applying these formal rules, we give an equational characterization for the 5-ary Bezout law determined by the conic process on the curve.

THEOREM . A 5-ary rational function f defined on a non-singular complex cubic is the same as obtained by the 5-ary conic process (i.e. $f = q$) iff f is symmetric in all its five arguments and satisfies the following two identities:

1. $f(E, E, E, E, E) = E$ for some inflexion point E on the curve.
2. $f(f(x, y, E, E, E), x, E, E, E)) = y$.

From this, we deduce an equational description of Bezout laws of higher arities, using the powerful Cayley - Bacharach Theorem on the intersection cycle of algebraic curves. It turns out that *all* these naturally defined operations of *higher* arities are just reducts of a single *binary* law of composition on the cubic curve!

R. Padmanabhan, University of Manitoba

Linked Double Weak Stone Algebras

by

Hanamantagouda P. Sankappanavar

Department of Mathematics
and Computer Science
State University of New York
New Paltz, New York

Department of Pure Mathematics
University of Waterloo
Waterloo, Ontario
Canada

ABSTRACT

An algebra $L = \langle L, \vee, \wedge, *, +, 0, 1\rangle$ with a bounded distributive lattice reduct is a linked double weak Stone algebra if the following conditions hold:

(i)	$0^* \approx 1$, $1^* \approx 0$;	$0^+ \approx 1$, $1^+ \approx 0$
(ii)	$(x \vee y)^* \approx x^* \wedge y^*$;	$(x \wedge y)^+ \approx x^+ \vee y^+$
(iii)	$(x \wedge y)^* \approx x^* \vee y^*$;	$(x \vee y)^+ \approx x^+ \wedge y^+$
(iv)	$x^{***} \approx x^*$;	$x^{+++} \approx x^+$
(v)	$x^* \wedge x^{**} \approx 0$;	$x^+ \vee x^{++} \approx 1$
(vi)	$x^{+*} \approx x^{++}$;	$x^{*+} \approx x^{++}$.

Let LDWS denote the variety consisting of all linked double weak Stone algebras.

In this paper we show that the variety LDWS has equationally definable principal congruences and give a complete description by Hasse diagrams, of the 27 non-isomorphic subdirectly irreducible algebra in LDWS.

Domains and Algebras

Dana S. Scott
Carnegie Mellon University

The theory of domains was introduced in the study of denotational semantics of programming languages in order to have structures appropriate for a variety of recursive definitions. It turned out that the domains themselves could also be defined by recursive definitions. These definitions through domain equations give structures with many kinds of algebraic operations. The talk will review the theory through a number of examples, including the construction of lambda calculus models. As algebras, lambda calculus models satisfy no non-trivial equations; however, according to a theorem of Engeler, these models contain arbitrary algebras as subalgebras. A simplified proof of the theorem will be presented together with some generalizations.

The Resolution Rule: An Algebraic Perspective

Zbigniew Stachniak
York University

In recent years a spectacular growth of work in computer science applications of non-classical logics has been accomplished. A number of important applications of such systems have been found in areas such as artificial intelligence, database theory, specification, verification, and synthesis of sequential and distributed systems. Various kinds of logics specially tailored for these disciplines have been studied, e.g. algorithmic, dynamic, and temporal logics, logics of knowledge and belief, etc. The central question in this logical direction is how the major problems can be transformed into a task of finding proofs of theorems, and whether natural and efficient automated theorem proving techniques for such tasks can be developed. Since their introduction in 1965, proof systems based on Robinson's rule (we shall call them *resolution proof systems*) have been extensively studied by the automated theorem proving community in the context of finding natural and efficient proof systems to support a wide spectrum of computational tasks. These systems, however, are mostly constructed and their properties studied in an *ad hoc* manner while there are theoretical grounds for the development of general techniques.

The goal of this paper is to show that propositional resolution proof systems can be conveniently introduced and studied within an algebraic framework. We view a resolution proof system as a finite universal algebra (of formulas) augmented with an inference operator. We prove a number of fundamental methodological results. Among them we show that all resolution counterparts of propositional logics are determined by a class of the so-called strongly finite logics. We also show that there is an effective procedure to generate a resolution proof system for every strongly finite logic.

Incremental Models of Incomplete Information Data Bases

Marek A. Suchenek

The Wichita State University
KS 67208

1 Abstract

We present a forcing-based model for a data base with incomplete information, where possible increases (increments) of its content are restricted to finite sets of atomic or negated atomic first-order sentences. Among others, we address the following problems related to such model:

- interpretation of negative information,
- reasoning from positive fragments,
- non-monotonicity of answers to queries,

applying model-theoretic forcing of [Rob71], described also in [Bar78], chap. A2 §8.

References

[Bar78] Jon Barwise, editor. *Handbook of Mathematical Logic.* North-Holland, Amsterdam, second edition, 1978.

[Rob71] Abraham Robinson. Forcing in model theory. In *Proc. Simp. Mat.*, pages 64–80, Institute Nationale di Alta Matematica, 1971.

Final Algebra Semantics for Insufficiently Complete Specifications

Satish R. Thatte
Department of Electrical Engineering and Computer Science

Lawrence S. Moss
Department of Mathematics

The University of Michigan, Ann Arbor, MI 48109

This paper presents a new approach to the semantics of algebraic specifications which removes the applicability restrictions of final algebra semantics while retaining its "most economical realization" property. The key idea involves lifting the question of semantics from individual theories to *languages* of theories based on a common signature and a common notion of externally observable behavior. The semantics of each individual theory is required to be *compatible* with the that of smaller and larger theories in the language. We illustrate the usefulness of these ideas in several fields, and we study the computability properties of final models of languages.

Richard Thompson, University of California

THE MANIPULATORY FOUNDATIONS OF NON-PARALLEL PROGRAMMING —OR, CAN ALGEBRAIC LOGIC GROW ON TREES?

An intensional analog to combinatory logic — attributory logic — is presented, and it is shown how it can be used to define the semantics of sequential programs. The manipulatory part of attributory logic gives rise in a natural way to automata on infinite binary trees — whose finitely-many generating operations can each be represented by a few assembly-level computer instructions (in fact, the constructive proof of manipulatory completeness can be turned into a practical program for a tree-based macroprocessor). It is easy to prove that the semigroup generated by the tree-manipulation operations has a finite presentation; using these operations to define higher-level transformational operations — and adding the usual Boolean operations together with a single tree-based quantifier — produces a finitely-based algebraic logic which can encode first-order logic.

ALL RECURSIVE TYPES DEFINED USING PRODUCTS AND SUMS CAN BE IMPLEMENTED USING POINTERS

Eric G. Wagner
Computer Science Principles
Mathematical Sciences Department
IBM Research Division, T. J. Watson Research Center
Yorktown Heights, NY 10598/ USA

Abstract:
We present a rigorous algebraic formulation and proof of the folk theorem to the effect that all the recursive types defined using "products and sums" (e.g., STACKs, TREEs, etc.) can be implemented using pointers. We give an algebraic formulation of recursive types and their operations. A collection of recursive types, together with their inherent operations, form an algebra. We give a algebraic formulation of imperative programming languages with pointers and variables. Roughly speaking, such a language corresponds to a collection of algebras with transformations between the algebras. Thus the two mathematical frameworks are rather different. What we show though is that "the usual" implementation of recursive types by pointers works, from a mathematical point of view, because, in a sense to be explained in the talk, it induces an algebra which is homomorphic to the desired algebra of recursive types.

A CHARACTERISTIC OF λ DEFINABLE TREE OPERATIONS

Marek Zaionc
Department of Computer and Information Sciences.
University of Alabama at Birmingham
University Station, Birmingham, Alabama 35294

Abstract. λ-language over simple type structure is considered. Type $\Upsilon = (O \to (O \to O)) \to (O \to O)$ is called binary tree type because of the isomorphism between binary trees and closed terms of this type. Therefore any closed term of type $\Upsilon \to (\Upsilon \to \ldots \to (\Upsilon \to \Upsilon)\ldots)$ represents a n-ary tree function. The problem is to characterize tree operations represented by the closed terms of examined type. It is proved that the set of λ definable tree operations is the minimal set containing constant functions, projections and closed for composition and recursion. This result should be contrasted with results of Schwichtenberg and Statman (cf. [Sch75],[Sta79]) which characterize the functions over natural number type $\mathbf{N} = (O \to O) \to (O \to O)$ and Zaionc (cf [Zai87]) for a word λ definable functions over type Type $\mathbf{B} = (O \to O) \to ((O \to O) \to (O \to O))$ by means of composition only.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Iowa State University, Department of Mathematics / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Ames, Iowa 50011	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Applications of Algebraic Logic and Universal Algebra to Computer Science

12. PERSONAL AUTHOR(S) Bergman, Clifford H.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 88-5-1 TO 89-4-30	89-6-21	27

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Data Types, Terms, Semantics, Domains, Specifications, Catagory Theory, Automata

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Iowa State University will host a four-day conference in June 1988, on the applications and methods of algebraic logic and universal algebra in theoretical computer science. Our primary goal is to bring together computer scientists and mathematicians working in this area to explore the interests and problems they have in common, and establish some avenues of communication.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE